# CIS11 Course Project Part 1: Documenting the Project

Fill in the following areas (purple).

**Introduction**

* 1. **Purpose**

The purpose of the program/code is to calculate the minimum , average and max of all 5 test scores inputted.

* 1. **Intended Audience and Users**

The audience can be from students to teachers using the program and as the users we m9ght end up using it to calculate what grade we need to pass a class.

* 1. **Product Scope**

The intention is to show I paid attention in class but also allow the professor to see what I can do and understand.

* 1. **Reference**

**Source Documents for the Program Requirements and Specification**

input: User is prompt to input the test scores.

Output: Display maximum, minimum, average scores and letter grade equivalence (0 – 50 = F,

60 – 69 = D, 70 – 79 = C, 80 – 89 = B, 90 – 100 = A) on the console.

**2. Overall Description**

**2.1 Product Perspective**

The primary code goal is for it to fully calculate and save everything we typed in to get our goal.

* 1. **Product Functions**

**The overall description of functionality:**

The main sub is to continue to run the program without having to reinitialize the machine.

* 1. **User Classes and Characteristics**

**What type of users are involved in this development process? Include business and technical personnel and their tasks.**

* 1. **Operating Environment**

What type of system will the application be operated on? Operating system? System types? Development platform?

How should the application be used? Simulator version?

* 1. **Design and Implementation Constraints**

Note any constraints or limitations to the application.

* 1. **Assumptions and Dependencies**

Note any dependencies.

Is this application dependent on other applications or services? Browser? Web services? Simulator?

***3*. External Interface Requirements**

* 1. **User Interfaces**

The user will be prompted to enter 5 numbers, and from there the program will do the rest of the work and output the results.

* 1. **Hardware Interfaces**

The user will only need a mouse, a keyboard, their computer, and a screen. The mouse to run the program, the keyboard for input, the computer to load the program, and the screen to see.

* 1. **Software Interfaces**

This software will only need a LC – 3 simulator to run.

* 1. Communications Interface

This application does not require any network connection. It only requires the code and a simulator to run the program.

**4. Detailed Description of Functional requirements**

**4.1     Type of Requirement (summarize from Section 2.2)**

**The functions are mainly ORIG., fill, array, input, and output. The purpose of the program is to show how a code can calculate a minimum, average, and max of 5 test scores. There are 5 inputs which are 52, 87, 96, 79, 61. The output should be the maximum grade followed by the average and minimum.**

**4.2 Performance requirements  
 I expect it to run with no issues at all.**

**4.3 Flow Chart OR Pseudocode**

**Pseudocode**

Set Origin at x3000

Start program

Output welcome to program

Load variable for a new line

Output

Start of functions

**Get grades**

Load variable for amount of grades

Clear registers

**Start loop function**

Clear registers

Load variable for translation

Get user input

Output input

Copy input

Convert input

Clear register

**Start of Multiplication function – to make number in the 10s**

Copy input to register

Decrement counter

Loop multiplication function until counter is zero

Get user input

Output user input

Convert input

Add to first number

Add space

Store value

Get letter grade for value

Add new line

Move to next value in array

Decrement counter

Back to beginning of loop if counter is positive

Add new line

**Start of calculating average function**

Load number of grades

Load array of grades

Clear registers

**Start of sum loop**

Add numbers until counter is zero

**Stat of Division**

Load number of grades

Get negative value of loaded number

Clear registers and load sum into another register

**Start of Division loop**

Subtract until negative result

Increment quotient counter

Loop until counter is zero

**Output**

Store average

Load string

Output string

Clear registers

St average in another register

Jump to subroutine CONVERT

Load average to register

Jump to subroutine LETTER

Add newline

**Calculate Max Function**

**Calc Max**

Load Number of grades

Load grades array

Load first grade into register

Store grade in max variable

**Max\_Loop**

Load next value in array

Load max value

Make negative for comparison

Compare array value and max

If max is less Branch

Else continue

Move to next element in array

Decrement counter

Continue loop if positive

Else continue to Minimum calculation

**Update\_Max**

Store new max value in max

Move to next element in array

Decrement counter

Continue to loop if positive

Else continue to minimum calculation

**Calculate Min Function**

**Calc Min**

Load Number of grades

Load grades array

Load first grade into register

Store grade in min variable

**Min\_Loop**

Load next value in array

Load min value

Make negative for comparison

Compare array value and min

If min is greater Branch

Else continue

Move to next element in array

Decrement counter

Continue loop if positive

Else continue to output

**Update\_Min**

Store new min value in min

Move to next element in array

Decrement counter

Continue to loop if positive

Else continue to output

**Output**

Load string

Output string

Ld max

Convert max to ascii character

Output

Get letter of the grade

Output new line

Repeat for minimum number

**Stop Program**

**Subroutines**

* **Subroutine for grades letter -**

**Letter**

Clear register

load value for a

Load grade for a

Compare value and input

If grade is an a branch to STR\_G

Repeat for grade b – f

Return to main function

**STR\_G**

Store save location

Clear register

Add grad to register

Output grade

Load store location return

* **Subroutine to convert decimal to ascii –**

**Convert**

Store save location

Load translation value

Copy value to be converted in register

**Div\_Loop2**

Increment quotient

Divide by 10

Repeat until negative

Subtract extra 1 from quotient

Add 10 to remainder in order to calculate second ascii character

Subtract from register 6

If positive or zero branch to POS

Else continue to negative

**Neg**

Increment quotient

Divide by 10

**Pos**

Store register 1 in quotient

Store register for in remainder

Load quotient, convert, output

Load remainder, convert, output

Output space

Load save location

Return to main function

**Data**

**End Program**

|  |
| --- |
|  |
|  |  |

|  |
| --- |
|  |
|  |  |